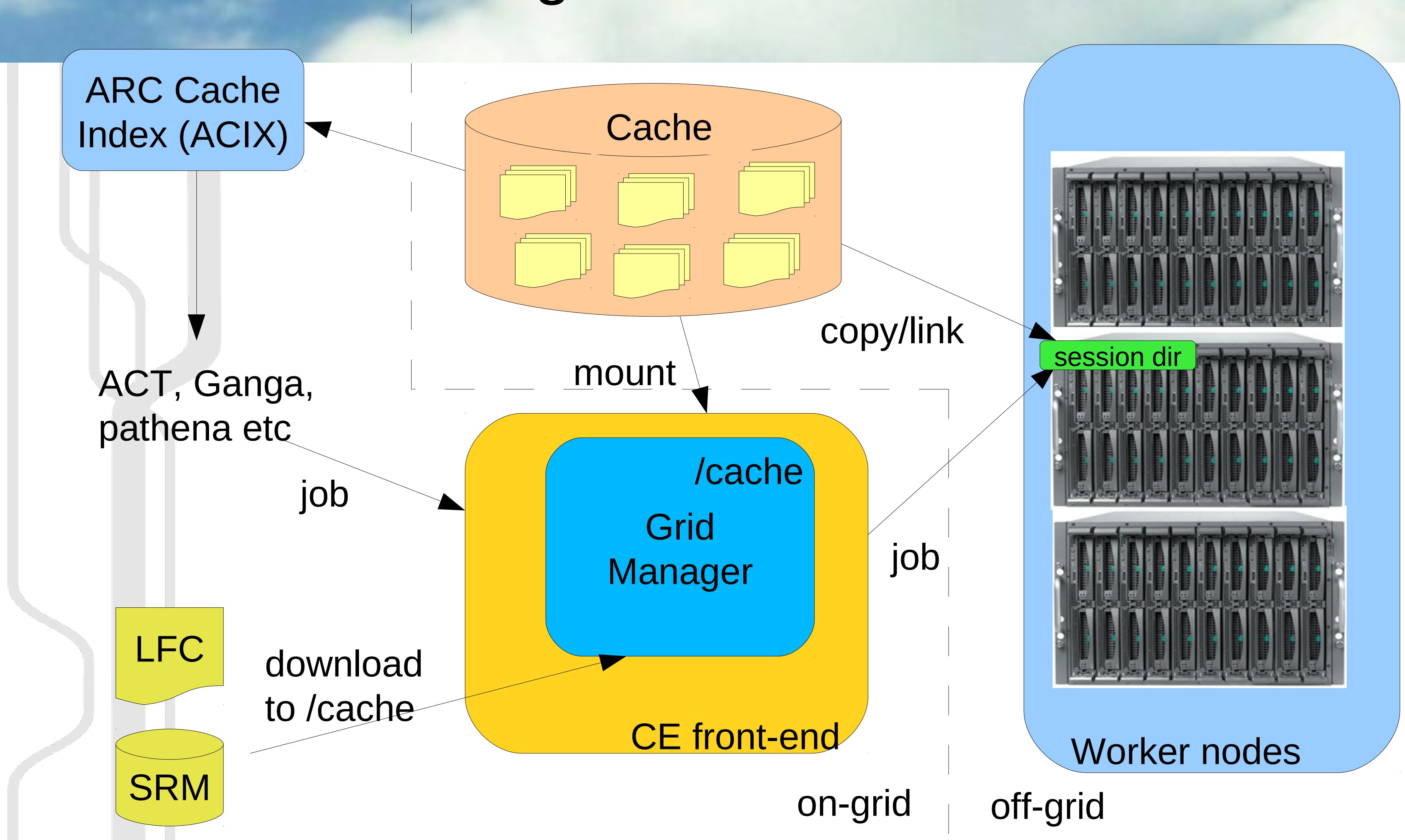
ARC Caching

David Cameron
University of Oslo/NDGF

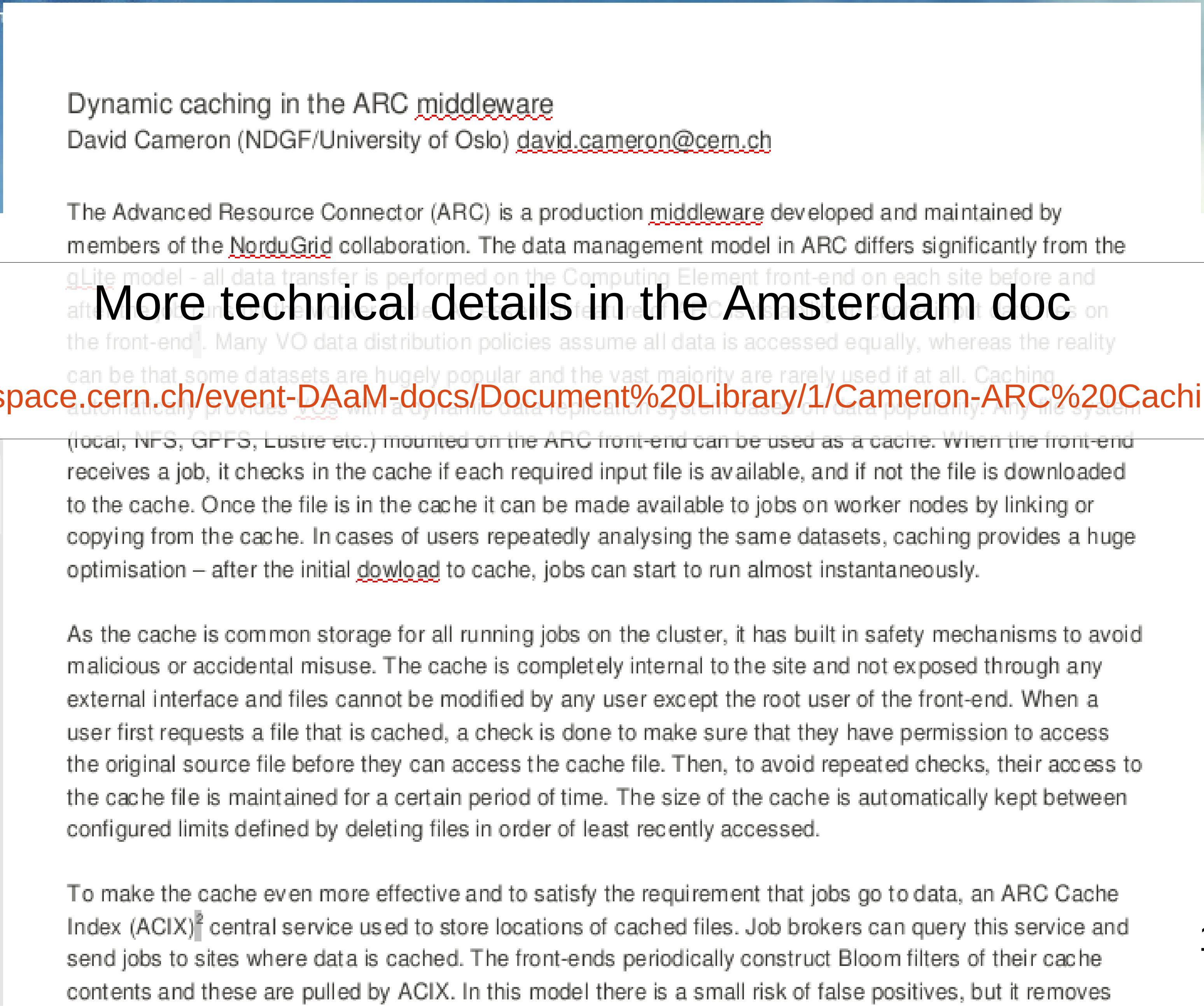
ARC Caching Architecture



Advantages of caching

- Less data transfer
- Jobs start faster
- No “SE” administration (LRA clean up, cache is internal)
- No consistency problems

= Less work!



Dynamic caching in the ARC middleware

David Cameron (NDGF/University of Oslo) david.cameron@cern.ch

The Advanced Resource Connector (ARC) is a production middleware developed and maintained by members of the NorduGrid collaboration. The data management model in ARC differs significantly from the gLite model - all data transfer is performed on the Computing Element front-end on each site before and aft [illegible] on the front-end[1]. Many VO data distribution policies assume all data is accessed equally, whereas the reality can be that some datasets are hugely popular and the vast majority are rarely used if at all. Caching automatically provides VOs with a dynamic data replication system based on data popularity. Any file system (local, NFS, GPFS, Lustre etc.) mounted on the ARC front-end can be used as a cache. When the front-end receives a job, it checks in the cache if each required input file is available, and if not the file is downloaded to the cache. Once the file is in the cache it can be made available to jobs on worker nodes by linking or copying from the cache. In cases of users repeatedly analysing the same datasets, caching provides a huge optimisation – after the initial dowload to cache, jobs can start to run almost instantaneously.

As the cache is common storage for all running jobs on the cluster, it has built in safety mechanisms to avoid malicious or accidental misuse. The cache is completely internal to the site and not exposed through any external interface and files cannot be modified by any user except the root user of the front-end. When a user first requests a file that is cached, a check is done to make sure that they have permission to access the original source file before they can access the cache file. Then, to avoid repeated checks, their access to the cache file is maintained for a certain period of time. The size of the cache is automatically kept between configured limits defined by deleting files in order of least recently accessed.

To make the cache even more effective and to satisfy the requirement that jobs go to data, an ARC Cache Index (ACIX)[2] central service used to store locations of cached files. Job brokers can query this service and send jobs to sites where data is cached. The front-ends periodically construct Bloom filters of their cache contents and these are pulled by ACIX. In this model there is a small risk of false positives, but it removes

More technical details in the Amsterdam doc

https://espace.cern.ch/event-DAaM-docs/Document%20Library/1/Cameron-ARC%20Caching.pdf

Some Statistics

- (Almost) all T2 storage in NDGF is ARC cache (1PB total)
- Recommended 100TB for ATLAS analysis – 2 week turnover under heavy load

Some index statistics (ATLAS)

Timespan	: 2010-06-23 - 2010-07-05
Queries	: 112391
N URLs	: 656669
Queries / Hour	: 390.51 (avg 10k jobs/day)
URLs / Query	: 5.84 (std. dev. 7.35)
Not cached URLs	: 10.56%
Cached URLs	: 89.44%
Hits / URL	: 3.06
Hits / URL %0	: 3.42 (excludes URLs with zero hits)

But...

- Integration takes a lot of work
  - VOs can be “difficult”
- Manpower is available for possible future changes
  - Pilot jobs?
  - Separation of cache?
- Plans
  - ATLAS NDGF cloud is in stable production
  - Experiment with other clouds
  - Other experiments?